Projektowanie Systemów Informatycznych

Inżynieria oprogramowania dotyczy stosowania empirycznego, naukowego podejścia do znajdowania wydajnych i ekonomicznych rozwiązań praktycznych problemów w dziedzinie oprogramowania.

Cykl życia systemu informatycznego – etapy i krótka charakterystyka:

Planowanie (identyfikacja celów), **Analiza wymagań** (określenie funkcji systemu), **Projektowanie** (architektura i struktura danych), **Implementacja** (kodowanie), **Testowanie** (weryfikacja poprawności), **Wdrożenie** (uruchomienie systemu), **Eksploatacja i utrzymanie** (monitorowanie, aktualizacje), **Wycofanie** (zamknięcie systemu).

Projektowanie systemów to proces definiowania architektury, komponentów, modułów, interfejsów systemu oraz danych dla systemu tak, aby spełniał on określone wymagania użytkownika końcowego. **Obejmuje ono „tłumaczenie" wymagań użytkownika na szczegółowy projekt, który kieruje fazą implementacji (obejmującej kodowanie). Celem jest stworzenie dobrze zorganizowanej struktury, która spełnia zamierzony cel, jednocześnie biorąc pod uwagę takie czynniki, jak skalowalność, łatwość utrzymania i wydajność.**

Znaczenie projektowania systemów w tworzeniu/rozwoju oprogramowania

Budowa udanej aplikacji wykracza poza posiadanie tylko użytecznych funkcji. Chodzi również o zapewnienie, że system przetrwa trudne warunki rzeczywiste, a więc jest solidnie przetestowany. Dobrze zaprojektowany system jest nie tylko niezawodny i skalowalny, ale także łatwy w utrzymaniu i rozwijaniu **w miarę zmiany wymagań**. Wraz ze wzrostem złożoności aplikacji, projektowanie ich z myślą o skalowalności, wydajności i dostępności staje się bardzo istotne.

Każdy system informatyczny/oprogramowanie wytwarza się etapami:

planowanie → analiza → projektowanie → implementacja

(etapów procesu tworzenia oprogramowania nie należy mylić z fazami cyklu życia systemu informatycznego, wówczas do powyższych dochodzą: testowanie, wdrożenie, eksploatacja, wycofanie)

Dlaczego warto uczyć się projektowania systemów?

W każdym procesie rozwoju, czy to oprogramowania, czy jakiejkolwiek innej technologii, bardzo istotnym etapem jest projektowanie. Bez fazy projektowania nie można przejść do implementacji lub testowania. Tak samo jest w przypadku systemu. Projektowanie systemów jest nie tylko kluczowym krokiem w rozwoju systemu, ale także reprezentuje logikę biznesową oprogramowania.

1. Planowanie

Jest to kluczowa faza w projekcie, gdy należy sobie odpowiedzieć po co tworzymy to oprogramowanie oraz jak ono powstanie. Na tym etapie są (powinny być) znane wyniki analizy biznesowej i studium wykonywalności – czyli powinna być podjęta decyzja, że oprogramowanie ma powstać. Tu ustala się: czy potrafimy to zrobić, czy projekt ma wartości biznesowe i jakie, czy będzie możliwe wdrożenie oprogramowania, jeśli powstanie.

(proces zarządzania projektem zaczyna się od opracowania harmonogramu prac i etapu analizy)

2. Analiza (wymagań)

Na tym etapie ustala się **kto** będzie używał systemu, **co** system ma robić oraz **gdzie i kiedy** będzie używany. To tu ma miejsce analiza i specyfikacja wymagań - system zostaje opisany wymaganiami konceptualnymi jako black box (czarna skrzynka), bez ustalania konkretnych technologii.

3. Projektowanie

To pierwsza decyzja o tym, jak system ma działać, jak ma to (tj. wymagania użytkownika) spełniać. Tu powstaje koncepcja wewnętrznej logiki systemu, jego architektura, interfejsy (w tym użytkownika). Obecnie stosuje się metody obiektowe[1], a projekt często stanowi abstrakcję, dość dokładną jednak, przyszłego systemu - jest to najczęściej forma opisu słownego w dokumencie, dodatkowo ewentualnie w postaci wizualizacji - rysunków, schematów, diagramów.

4. Implementacja

To końcowa faza procesu tworzenia oprogramowania. Podczas implementacji zostaje zbudowana baza danych w wybranej technologii implementacyjnej i wytworzony kod aplikacji.

(tutaj się kończy projektowanie/wytwarzanie oprogramowania, ale w zgodnie z fazami cyklu życia systemu informatycznego, po implementacji dochodzi: testowanie, wdrożenie, eksploatacja, wycofanie. Czyli po wykonaniu testów systemu można wdrożyć system w warunkach rzeczywistych i sukcesywnie utrzymywać go w ciągłej pracy, aż do jego finalnego wycofania).

[1] Programowanie strukturalne a obiektowe. Programowanie strukturalne skupia się na pisaniu funkcji. W programowaniu obiektowym najważniejsze są dane.

Cykl życia systemu informatycznego

Etapy, które prowadzą inżynierów przez proces tworzenia systemu zgodnego z potrzebami użytkownika i celami organizacji, mając na celu zapewnienie niezawodności, skalowalności i możliwości konserwacji produktu końcowego.

Wyobraź to sobie jako przepis na zrobienie naleśników. Zaczyna się od zaplanowania, jakiego rodzaju naleśniki chcesz przygotować (słodkie? słone? fluffy pancakes?), zebrania odpowiedniego przepisu i określenia sekwencji czynności, zgromadzenia składników, wymieszania ciasta, usmażenia naleśników, sprawdzenia, czy mają oczekiwany smak, a na koniec podzielenia się nimi z innymi, a następnie schowania do lodówki tego, który został niezjedzony (wycofanie może być jego wyrzuceniem do kosza).

Podobnie Cykl życia systemu pomaga specjalistom w rozwoju systemów, zapewniając jasny plan działania – od pierwszego pomysłu na system, przez jego stworzenie i wdrożenie, aż po jego bieżącą konserwację i ostateczne zamknięcie.

Cykl życia systemu informatycznego – etapy i krótka charakterystyka:

Planowanie (identyfikacja celów), **Analiza wymagań** (określenie funkcji systemu), **Projektowanie** (architektura i struktura danych), **Implementacja** (kodowanie), **Testowanie** (weryfikacja poprawności), **Wdrożenie** (uruchomienie systemu), **Eksploatacja i utrzymanie** (monitorowanie, aktualizacje), **Wycofanie** (zamknięcie systemu).

Etap 1. Planowanie

- Tutaj zaczyna się projekt! Ty decydujesz, co chcesz osiągnąć (cele), ile możesz wydać (budżet) i kto będzie nad tym pracował (zespół).
- **Przykład:** Firma planuje stworzyć system zarządzania klientami CRM, opisuje jego najważniejsze cechy i zespół zaangażowany w jego wdrożenie.

Etap 2. Analiza wymagań

- Przed rozpoczęciem sprawdź, czy pomysł jest praktyczny i przystępny cenowo? **Kto** będzie używał systemu, **co** system ma robić oraz **gdzie i kiedy** będzie używany?
- **Przykład:** Firma sprawdza, czy zbudowanie systemu CRM jest warte kosztów i wysiłku oraz czy może on przynieść oczekiwane korzyści i czy spełnia oczekiwania użytkowników rynku.

Etap 3. Projektowanie

- Na tym etapie należy zaplanować, jak system będzie działał i jak będzie wyglądał dla użytkowników.
- Wyobraź sobie projektowanie domu, zanim rozpocznie się budowa, architekci tworzą plany, które pokazują i opisują, gdzie będzie przebiegać każdy pokój, rura i przewód. Podobnie w projektowaniu systemu tworzy się szczegółowy plan, którego deweloperzy będą przestrzegać, aby zbudować system.

Etap 4. Implementacja

- Przekształć projekt w działający system. Programiści piszą kod, aby stworzyć system na podstawie projektu.
- **Przykład:** Kodowane są funkcje systemu CRM, takie jak profile klientów i pulpity nawigacyjne.

Etap 5. Testowanie

- Sprawdź, czy system spełnia określone wymagania i czy wszystko działa zgodnie z planem.
- System CRM poddawany jest różnym procedurom testowym, takim jak testy jednostkowe, testy integracyjne i testy akceptacji użytkownika, aby zapewnić jego funkcjonalność, wydajność i bezpieczeństwo.

Etap 6. Wdrożenie (uruchomienie systemu na produkcji, deployment/release)

- Cała ciężka praca kończy się w fazie uruchomienia, kiedy system jest udostępniany do rzeczywistego użytku. Po zakończeniu całego planowania, projektowania i budowania jest to podobne do uruchomienia nowego sklepu.
- Etap ten obejmuje przeniesienie systemu ze środowiska testowego lub programistycznego do środowiska produkcyjnego, w którym będą mogli uzyskać do niego dostęp faktyczni użytkownicy.

Etap 7. Eksploatacja i utrzymanie (konserwacja)

- Zapewnij ciągłą funkcjonalność i rozwiąż wszelkie pojawiające się problemy.
- Regularne aktualizacje, poprawki błędów i wsparcie użytkowników systemu w celu dostosowania go do zmieniających się wymagań biznesowych i rozwiązywania pojawiających się problemów.

Etap 8. Wycofanie

- Zamknięcie systemu.

Różnice pomiędzy cyklem życia systemu informatycznego a cyklem życia projektu systemu

Przeanalizuj różnice:

Aspekt	**Cykl życia systemu informatycznego**	**Cykl życia projektu systemu**
Definicja	Kompleksowe ramy obejmujące cały proces rozwoju systemu.	Podzbiór zajmujący się konkretnie projektowaniem systemu.
Zakres	Obejmuje cały cykl życia systemu, od planowania, przez uruchomienie do wycofania.	Koncentruje się przede wszystkim na aspektach projektowych systemu.
Fazy	Zazwyczaj obejmuje: planowanie, analizę wymagań, projektowanie, implementację, testowanie, wdrożenie, eksploatację, wycofanie.	Zwykle obejmuje: planowanie, analizę wymagań, projektowanie, implementację.
Centrum	Szerokie skupienie na całościowym procesie rozwoju, obejmujące planowanie, wdrażanie, testowanie i konserwację.	Szczególny nacisk kładzie się na fazę projektowania, szczegółowo opisując sposób budowy i działania systemu.
Zamiar	Przeprowadza zespół programistów przez cały proces – od koncepcji po wsparcie powdrożeniowe.	Zawiera plan budowy systemu w oparciu o określone wymagania projektowe.

Wyzwania w cyklu życia projektu systemu

- Czasami początkowe wymagania dotyczące systemu mogą być niejasne lub niejednoznaczne, co może utrudniać jego dokładne zaprojektowanie.
- Wymagania mogą ulegać zmianom w trakcie procesu projektowania, co stwarza wyzwanie w zakresie zachowania spójności i zapewnienia, że system nadal spełnia potrzeby użytkownika.
- Szybki postęp technologiczny może utrudniać wybór najwłaściwszych i najnowocześniejszych technologii do projektowania systemów.

- Zapewnienie płynnej integracji różnych komponentów systemu może być skomplikowane, szczególnie w przypadku stosowania różnych technologii i platform.
- Zaprojektowanie systemu w ramach ograniczeń budżetowych może okazać się trudne, ponieważ włączenie niektórych funkcji lub technologii może okazać się opłacalne.

Modele używane w cyklu życia projektu systemu

- **Model kaskadowy:** liniowy i sekwencyjny model, w którym każda faza musi zostać ukończona przed przejściem do następnej. To proste podejście, ale może być nieelastyczne w obliczu zmieniających się wymagań.
- **Model iteracyjny:** obejmuje powtarzające się cykle, przy czym każda iteracja udoskonala i ulepsza system na podstawie informacji zwrotnych. Jest dostosowywalny do zmieniających się wymagań.
- **Model prototypowania :** Polega na zbudowaniu prototypu (wersji wstępnej) systemu w celu zebrania opinii i dopracowania projektu przed zbudowaniem finalnego produktu.
- **Model spiralny:** Zawiera elementy modeli iteracyjnych i prototypowych.
- **Model Agile (zwinny):** Kładzie nacisk na elastyczność i współpracę, z częstymi iteracjami i ciągłym sprzężeniem zwrotnym. Dobrze nadaje się do projektów, w których wymagania mogą ewoluować.

Postępując zgodnie z najlepszymi praktykami i używając odpowiedniego modelu, firmy mogą zwiększyć swoje szanse na pomyślne ukończenie projektów rozwoju systemów.

Najlepsze praktyki w cyklu życia projektowania systemów

- Poświęć czas na dokładne zrozumienie i udokumentowanie wymagań, aby zapewnić solidną podstawę dla procesu projektowania.
- Utrzymuj otwartą komunikację z interesariuszami, aby mieć pewność, że ich potrzeby są zrozumiane, i aby móc szybko reagować na wszelkie zmiany.
- Projektuj systemy w sposób modułowy, umożliwiający łatwiejszą konserwację, aktualizacje i skalowalność.
- Identyfikuj potencjalne ryzyka już na wczesnym etapie procesu projektowania i opracowuj strategie ich łagodzenia lub skutecznego zarządzania nimi.

Przykłady zastosowań cyklu życia projektu systemu

- Opracowywanie nowych aplikacji programowych
- Udoskonalanie istniejących aplikacji programowych
- Integrowanie różnych systemów
- Wymiana starszych systemów
- Opracowywanie rozwiązań dostosowanych do konkretnych potrzeb biznesowych

Projektowanie nowoczesnych systemów informatycznych wspierających analizę danych i procesy decyzyjne oparte na algorytmach data science

Cykl życia procesu analizy danych w data science

1. Zdefiniuj cel (Jaki problem staram się rozwiązać?)
2. Zgromadź dane i zarządzaj nimi (Jakie informacje są mi potrzebne?)
3. Zbuduj model (Znajdź w danych wzorce prowadzące do rozwiązań)
4. Oceń model i poddaj go krytyce (Czy model rozwiązuje mój problem?)
5. Zaprezentuj wyniki i udokumentuj je (Udowodnij, że możesz rozwiązać problem i pokaż, jak tego dokonasz)
6. Wdróż model (Wdróż model tak, aby rozwiązywał problem w środowisku produkcyjnym. Wdrażanie i utrzymywanie modelu)

Bibliografia

- Farley D., Nowoczesna inżynieria oprogramowania. Stosowanie skutecznych technik szybszego rozwoju oprogramowania wyższej jakości, Helion 2023
- Śmiałek M., Rybiński K., Inżynieria oprogramowania w praktyce. Od wymagań do kodu z językiem UML, Helion 2024
- Wrycza S., Maślankowski J., Informatyka ekonomiczna. Teoria i zastosowania, PWN 2019
- Zumel N., Mount J., Język R i analiza danych w praktyce, Helion 2021